Continuous short-term forecasting of prices in financial markets

Alperovich E.[1], *Alperovich A.* [2]

1).*ALPIR Inc, New York, USA*
E-mail: ejen1@yahoo.com
2) *VOZZI Media LLC, Florida, USA*
E-mail: abivera.alperovich@gmail.com

Abstract: *The paper describes in detail and justifies the principles used to build a price forecasting system. Forecast of the upper and lower price bounds and the price itself, as well as the probability of the direction of the price trend.*

Keywords: stock market, trajectory matrix, Lagged-Vectors, current column, Hash Code, similarity coefficient, patterns, correlated columns.

1. Introduction

The forecast of prices on the securities market is crucial when making investment decisions in trading on the stock exchange [1].

Exchange behavior is usually represented as a time series of issuer prices. The simplest form of representation of such a time series is the equation:

$$y(t) = f(t) + \varepsilon(t) \qquad (1.1)$$

$f(t)$ is the component of this series that is the chosen time interval, and $\varepsilon(t)$ is a placeholder for the stochastic parts in the calculations. A *trend* is usually understood as the function *y(t)* "cleared" random components, i.e. *f(t)*.

Highlighting a trend is identifying a smoothly changing global model of series behavior. It is assumed that this component is "inertial"[1]. All the main forecasting methods are based on this assumption.

In forecasting, a considerable amount of attention is paid to the short-term forecast, which allows one to quickly respond to unexpected changes. Financial forecasting, including short-term forecasting, is based on the analysis of past data and uses various smoothing methods.

Trend selection is the task of smoothing the series by explicit and implicit methods. Implicit methods include the use of artificial neural networks, which have become popular recently [6]. Traditional forecasting networks consist of the same type of elements - neurons connected to each other. The network must first be

[1] This means that it will retain its behavior in the near future.

"trained" on examples from the history of time series. By strengthening or weakening the connections between neurons, the network is forced to perform an accurate extrapolation of the function using many variables describing the market.

There are classical smoothing methods that have not lost their relevance such as:

- Correlation methods which are based on finding the relationship between two or more selected values [4] [7].
- Regression algorithms [2] [9], in which the choice of a function is of great importance, assuming that it will be stored further for a certain time interval.
- Technical analysis, however, remains one of the most popular forecasting methods [8]. It uses similar price changes from the past to predict future price behavior and trends. In technical analysis, a significant set of axioms, criteria, and various techniques have been developed that are often not proven. They allow you to make heuristic conclusions about the future price behavior based on a quote chart, trading volume, and news information [11].

Below is a proposal for a different approach to predicting the future values of a random time series based off an analysis of its present and past values. A discrete series of prices on the exchange is considered, which are fixed at specific and uniform points in time.

With the chosen quantum of discreteness and a given step the forecast time interval, the price, as well as a few narrow ranges of its possible fluctuations, is predicted in advance for any point on the time axis. The principles of building a forecasting system and making decisions on a transaction are described and expanded on in detail.

Currently, a forecast of price behavior is made with simpliflied exponential smoothing. This method allows you to compare and additionally evaluate the results of forecasting.

The work uses only the widely accepted statement of technical analysis: “History repeats itself”. The axiom, in this context, is the conclusion that events occurring in the securities market are patterns and will eventually be replicated [8]. Inarguably, most market participants act in accordance with previously acquired experience and a set precedent, which causes said repetition of situations under similar external conditions. This behavior, however, does not automatically confirm that they are completely identical. The pattern merely confirms how price reactions are very similar, despite the different price sales, rise values, and fall values.

2. Preliminary data processing

Let's assume that the necessary forecasting step is selected and suppose that there is a time series of historical data[2]. The series contains a sufficient number of values (more than 100,000 points) over the specified period of time. Within this collective, we use a time series containing the closing prices of the intervals[3]. One is able to transform the series into a working file of length N, entering into it only the values with the selected step.

Working with the traditional method, consider a "sliding window" of width m, which moves through the working file in the direction of increasing the time by one (1) step.

This creates arrays of data $[x_i \ldots x_{i+m-1}]$, where i vary from 1 to $(N\text{-}m+1)$. The set of these data forms a so-called trajectory matrix [3]. All matrix elements are known quotation prices with the selected step, and the columns are the contents of consecutive windows.

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_i & \cdots & x_{N-m+1} \\ x_2 & x_3 & \cdots & x_{i+1} & \cdots & x_{N-m} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_m & x_{m+1} & \cdots & x_{i+m} & \cdots & x_N \end{bmatrix} \qquad (2.1)$$

The columns have a length m ($m << N$) equal to the width of the window and are called *lagged vectors*.

Consider the column corresponding to the current time $X_{N+1} = [x_{N-m+2} \ldots x_{N+1}]$. The column is not included in the matrix and contains an unknown price value x_{N+1}. We will refer to it as the *current vector* or *column*. The value of x_{N+1} is the price relative to the moment that will occur at the end of the next step $(N+1)$. Our goal is to calculate and predict this price.

To obtain quantitative estimates, we will try to select lagging vectors similar[4] to the current column in the trajectory matrix.Note that, by creating a trajectory matrix, we have made the time series transition to a vector representation. Additionally, we have excluded time dependencies, since the time parameter is not explicitly present in the matrix. This means that, after the transformation of the matrix, it is relatively impossible to use the generally accepted weight coefficients depending on the proximity of prices to the current moment.

---

[2] A sequence of quoted prices of a financial instrument received from the exchange at regular intervals (a discreteness quantum).

[3] The prices corresponding to the discreteness quantum end.

[4] KEY NOTE: Columns that are completely and positively correlated with each other will be called similar.

3. **Hash code development**

A short-term forecast requires a quick reaction to price changes. To reduce the number of further calculations and condense the work, it is more efficient to reduce the dimension of the trajectory matrix containing thousands of columns, leaving only similar columns in it.

To find similar columns, we build a special hash code. It is created it so that the rise or fall of the financial instrument price at each subsequent time point in comparison with the previous one changes the value of the hash code. For this purpose, each point of the column is assigned a binary code[5] corresponding to the sign of the first difference between prices at points: *i* and (*i*-1). The column has *m* points, so each lagged vector will receive a code equal to the decimal equivalent of the binary code of dimension *m*. The exact magnitude of the price rise/fall is not significant [12].

Note that with a column length (window width) $m = 3$ points, the number of similar columns is hundreds or even thousands; and for $m = 10$, the number of such columns in most cases is zero (0), even with a significant amount of historical data. Therefore, the actual column length (m) should not exceed six to seven (6-7) points. When the column length is $m = 7$ points, the highest decimal value of the most significant bit is 64[6]. This makes it possible to modify the hash code (for columns of length $m \le 7$) so as to split the areas of growth, fall, and constancy of prices (growth-hundreds, fall-units and price preservation hundredths).

A significant amount of noise is present in the stock market data. This noise is caused by a small number of random transactions, the difference in the fixing time, the prices of the instrument, delays in getting last trade, the difference in bid-and-ask information, and other factors. The generated hash code will partially eliminate the influence of the noise component. To do this, we will introduce a dead zone (Δ) into it for small price deviations. After taking into account all the listed changes, we have the absolute value of the hash code with three possible values:

$$|h(i)| = \begin{cases} 100\times 2^{i-1} & if \quad price(i) - price(i-1) > \Delta \\ 0.01\times 2^{i-1} & if \quad |price(i) - price(i-1)| \le \Delta \\ 1\times 2^{i-1} & if \quad price(i) - price(i-1) < -\Delta \end{cases} \tag{3.1}$$

i is the bit number in the column, price(i) is the price at the point i of the column

[5] Binary code consists of ones (1) and zeros (0).

[6] The count will begin at zero (0).

The hash code we have created may determine the direction of the price behavior at the point (m+1), which is not included in the column[7]. Taking into account the maximum value of the hash code module (even with a window width of 7 points, it does not exceed 12,800) in the work, as an equivalent of such a "sign", it is accepted:

- When the price increases at the point (m+1), the absolute value of the hash code increases by two hundred thousand (200,000).
- When the price goes down, the absolute value of the hash code increases by one hundred thousand (100,000), and the hash code is in the range of 100,000 to 200,000,
- If the price does not change, the hash-code of the vector is equal to its modulus.

$$h(i)=\begin{cases} |h(i)|+200,000 & if \quad rice(m+1)-price(m) > \Delta \\ |h(i)| & if \quad |price(m+1)-price(m)| \le \Delta \\ |h(i)|+100,000 & if \quad price(m+1)-price(m) < -\Delta \end{cases} \tag{3.2}$$

The hash code splits the trajectory matrix into three compressed matrices: one matrix for the price increase after each column (point "m+1"), the second for the fall at the same point, and the third when the price change is less than or equal to Δ. The use of the specified hash code allows you to repeatedly reduce the dimension of the obtained matrices, leaving them only similar columns with the same absolute value of the hash code equal to the modulus of the hash code of the current column.

4. Scaling

Prices in different columns of the three obtained matrices are correlated, but their values may differ significantly from one another. Each column of the matrix is a function defined by m points. These functions defined in each column are realizations of the process describing price changes at different time periods.

Therefore, these functions should differ from one another and from the function of the current column due to changes in external conditions. To compare

[7] It should have three different values for the growth, fall and preservation of the price value at this point (some equivalent of a sign).

the columns with each other, it is necessary to scale them in accordance with the prices in the current column [5].

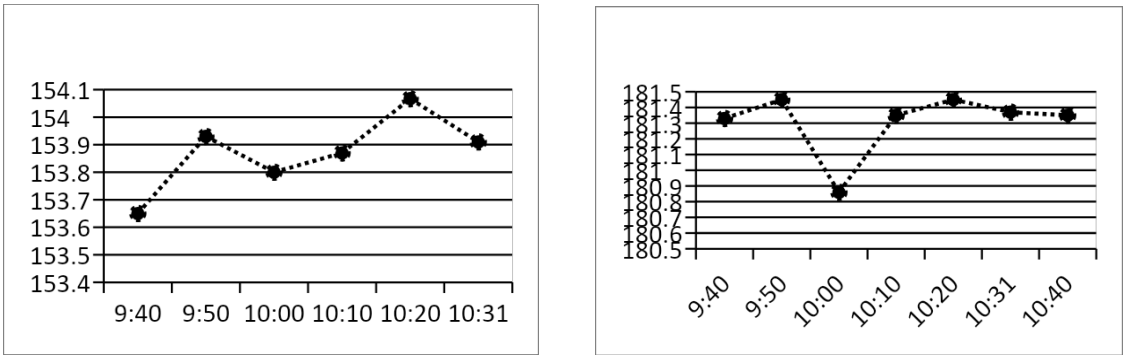

Fig. 1 *IBM Current vector and correlated vector. Step =10.*
Prices (≈$154 and ≈ $181) at 10: 30. Their values and deviations differ.
The price (U) at the point N+1 (10:40) for the current vector (N=m) is not known.

If the columns are considered similar, then a constant similarity coefficient should be introduced. We will get scaled matrices in which the full correlation of columns will be preserved. Each column '*j*' will have its own coefficient a_j.

After modifying the columns of the matrices by multiplying by the corresponding factors, we obtain matrices in which the columns can be compared with each other and with the current vector.

Note that a more accurate approximation, such as a linear one, slightly improves accuracy, but distorts the correlation of the columns.

5. Evaluation of the price

The columns in the matrices after scaling are similar to the current column, but are described by different functional dependencies.

The true price before the end of the one-step time interval is unknown. However, the number of columns in the matrices are known (K_0 – for the hash code ≤, |Δ|, K_- - for the hash code < -Δ and K_+ - for the hash code > Δ). The corresponding prices at points (m+1) for all modified vectors (patterns) in each of the matrices are also known.

Therefore, the weighted average values of prices for all patterns at the point (m+1) can be taken as the expected price.

$$Exp\_price[N+1]=\frac{\sum_{K_0} exp\_prc(0)+\sum_{K_-} exp\_prc(-)+\sum_{K_+} exp\_prc(+)}{K_0+K_-+K_+} \qquad (5.1)$$

where Exp_pice is the expected price at (N+1) point of the current vector,
exp_prc - "0", "-", and "+" are the prices of patterns at (m+1) points.

It is also possible to estimate the probability of the direction of the price trend at the point (N+1) [11].

$$p_+ = \frac{K_+}{K_+ + K_- + K_0}$$
$$p_- = \frac{K_-}{K_+ + K_- + K_0}$$
$$p_0 = 1 - p_+ - p_- \qquad (5.2)$$

p_+ , p_- and p_0 – are the probabilities of the price rising, falling or maintaining at the point $x_{N.+1}$.

If any matrix is empty, or no patterns corresponding to the current vector are found, then there will be a hash-code <100,000. The search for the expected price does not make sense since the price does not change and is equal to the current price.

With a matrix that is empty and a hash-code >100,000, it is impossible to determine the expected price from patterns. In this case, the expected price is assigned a value obtained by linear approximation based on all points of the current vector by the least squares method.

6. Boundaries fluctuations of the price

There are various approaches to assessing the boundaries of price fluctuations; one of which is the use of the nearest patterns.

Among all the selected modified vectors, we will find the vector closest in behavior to the current vector.[8]

Prices at points (m+1) for both nearest patterns (for price growth and its fall) are known. It is natural to assume that the price of each nearest pattern at the (m+1) point will be an estimate of the expected price.

Estimates based on the nearest patterns form a relatively narrow tube of deviations from the real price, and are used in the program.

Charts of the nearest patterns and the price behavior at the local minimum point are shown in Fig. 2 and Fig. 3.

[8] KEY NOTE: The closest vector is the one whose maximum modulus of deviations from the current vector at all points of the column is minimal.

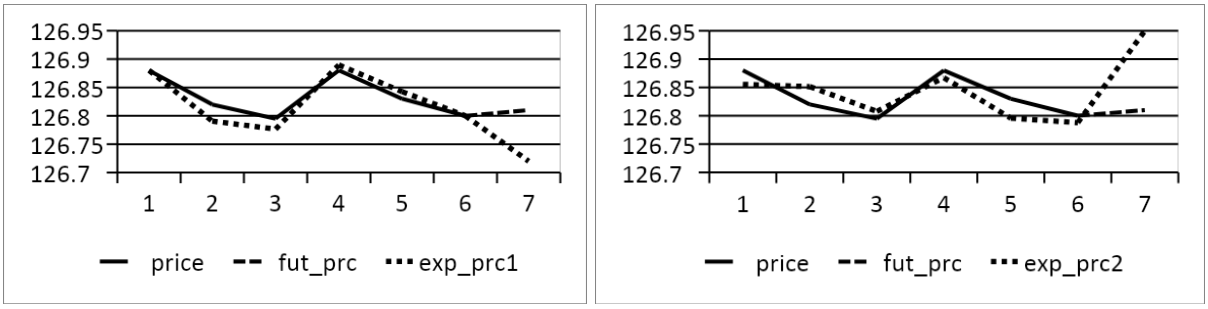

Fig. 2. *AAPL. The nearest patterns, both for the fall and for the growth of the expected price. Window = 6 points. Time 11:21, step = 5 min. The point is the actual future price at the 7th point. Expected price fall - pattern # 137. Growth of the expected price pattern # 124*

To obtain a more accurate boundary forecast of price fluctuations, taking into account noise, it is necessary to expand estimates based on the nearest patterns by 0.01 – 0.02%.

Note that similar results were obtained for all related criteria, both absolute and standard deviations of patterns from the current vector at all points of the column, as well as at the last three (3) points. Additionally, consider that estimating the boundaries is not related to getting the expected price.

The estimates obtained, shown in Fig.3, are conditional boundaries, which mean that the real price may exceed them, no matter how unlikely.

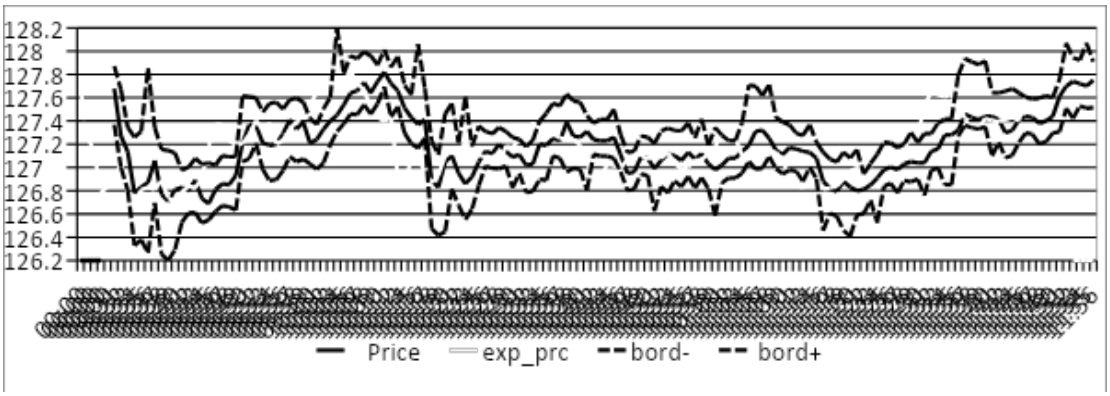

Fig.3 *Price and expected boundaries of its fluctuations for AAPL from 12/16/2020 The price reaches a local minimum at the time of 11:21 (see Fig.2).*

The boundaries of fluctuations in the expected price outwardly resemble the well-known indicator "Bollinger Band" [10], although they have nothing in common with it. They allow you to assess the behavior and volatility of prices not after the fact, as in the case of the "Bollinger band", but as a step forward.

The price forecast and its boundaries appear simultaneously with the corresponding values of the current price, as the delay for calculations is extremely

small. However, on the combined "price-forecast" chart, the points and boundaries of the price forecast are shifted to the right by one step relative to the current price.

7. Some features of the forecast program.

The program uses a window of 6 points, and the quantum of discreteness is one minute.

At the beginning of the program, the historical file is expanded with the previous day's data to reduce the price “jump” between the current opening price and the long-standing closing price stored in the file. Next, it is converted into a working program file with a specified step and a corresponding hash code file. All these preliminary operations are performed before the start of forecasting, so they do not require time during the calculation of forecast values.

The working file and the hash code file in the process should be modified by adding the data of the current day to them.

Allow us to elaborate:

Suppose, for example, that when the exchange opened, there was a significant drop in price compared to the closing price of the previous day entered in the working file. The current-day price should not exceed this closing price during the entire trading period. If no new data is entered in the working information, then the calculation of any hash code will use (m-1) prices values of the previous day and the price of the current day, which is always less than the closing price, i.e. the hash-code will not change.

When the first price of the day is received, the current vector is filled with the prices of the previous day. They are considered as the prices of the day, despite being received earlier. Usually there is a ”jump" between the closing price of the previous day and the current day opening price, which creates an additional forecast error. In the future, the current vector will gradually be filled with new prices, and the number of prices of the previous day will decrease, therefore, the forecast error caused by this jump will decrease.

The program uses working arrays of historical data with any given step. The same step should be saved in the current vector when it is filled with the data of the current day. Since in the current vector, the data of the day step-by-step replaces historical data, after six (6) steps (the width of the window in our case is 6 points), historical data will completely disappear in the vector. In this case, the vector will contain six prices of the day, and the forecast will give the expected price value at the point one step ahead. In a minute (1 quantum of time), a new price value will come. New prices of the day, shifted by one (1) quantum, will be entered into the current vector, since the forecast is carried out at the moment corresponding to the next price, shifted by a minute. The process will repeat.

The described process will be repeated all the time until a signal is received about the end of the day's data.

After each step specified by the user is completed, the working array of historical data is updated with a new value. The working arrays of historical data within the step do not change.

If there are gaps in the data coming from the input[9], the program will fill in the gaps by interpolation.

At each moment of time, three possible hash-code values are considered, corresponding to the preservation, fall of the price, and rise of the price. The necessary calculations are performed for each of them. The described algorithm is executed for any new price of the current day.

In order for the boundaries of the expected price fluctuations to be smooth and more convenient to use, it is desirable to partially smooth them out. The program uses border smoothing by averaging over three (3) points.

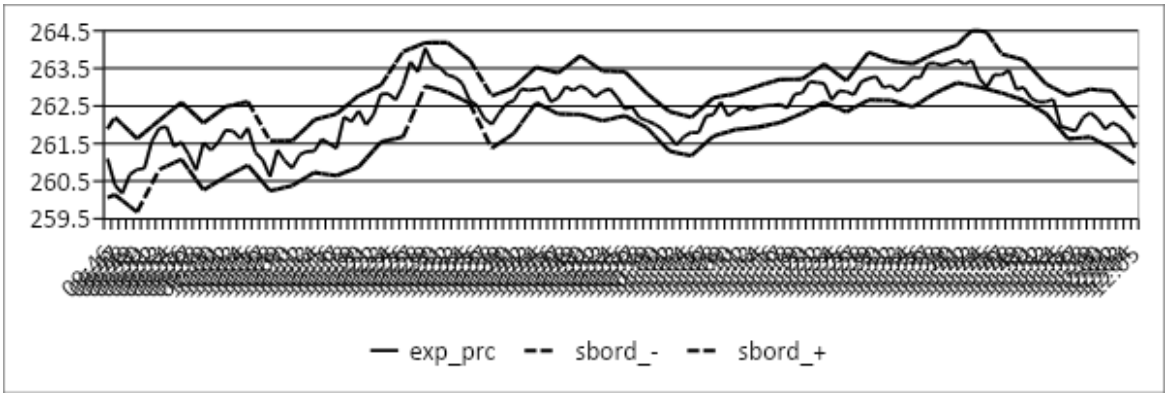

Fig. 4 *MSFT from 01/03/2023 step 5 min*

The resulting forecast, like any forecast, can give an error. To confirm the results, it is desirable to obtain a forecast of the expected prices by an independent, fundamentally different method. For this purpose, for example, the widely used method of simple exponential smoothing is suitable, which is additionally included in the program [12].

[9] Any time points are missed.

8. Exponential smoothing

Usually, simple exponential smoothing with a one-step forecast is performed using the well-known Brown formula:

$$S_{t+1} = \beta \times y_t + (1-\beta) \times S_t, \qquad (8.1)$$

S_{t+1} is the predicted value, y_t is the current value of the series, S_t is the previous forecast, and β is the smoothing factor (0< β <1).

The value of β determines how quickly the weight of the prices preceding the current one decreases during the forecast. If $\beta = 1$, there is no smoothing at all and the resulting series completely repeats the original one.

The first price value ($S_t = y_0$) of the day was selected as the initial forecast value (S_t) in the program. Then, for any β, $S_1 = y_0$ (equivalent to a shift to the right), i.e. the forecast is equal to the first price of the day (counting from 0). The resulting initial prediction error decreases rapidly with an increasing number of steps.

The value β, like other parameters, is specified in the program configuration file.

If the step is the same, the forecast obtained by exponential smoothing allows you to confirm the result of the previous forecast, or at least the direction of the trend. The graph of the exponential smoothing forecast is displayed simultaneously with the forecast described above. To combine the graphs of the proposed and exponential forecasts, in the latter linear interpolation is performed within each step between the predicted points.

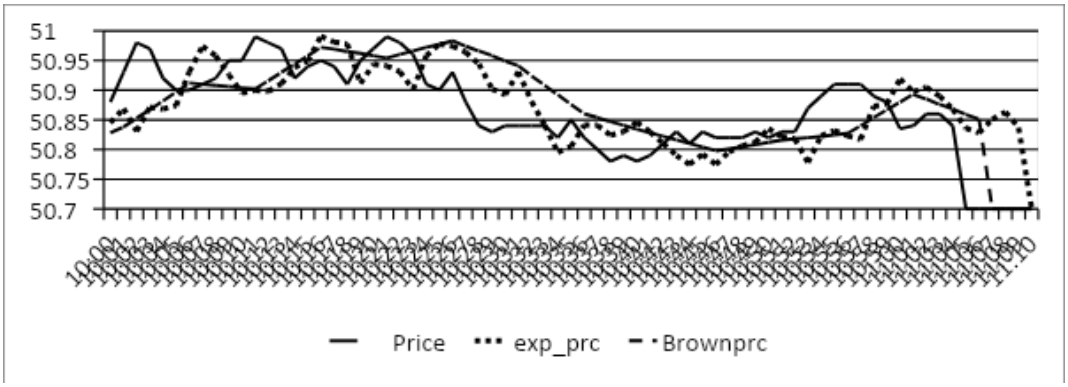

Fig. 5 *INTC from 12/16/2020 step 5 min.*

Unlike the forecast described above, with exponential smoothing, the predicted values appear only at the moment corresponding to the specified step, i.e. the forecast on the graph is performed in “jumps”. Linear interpolation within each step provides smoothing of this graph.

Conclusion

Keep in consideration that in the described program, three independent forecasts are actually implemented; in addition to exponential smoothing and the price forecast based on the average weighted values of all patterns. The price forecast is represented by a median line of boundaries, obtained from the nearest patterns, having no correlation or resemblance with the specified forecasts.

All the main parameters of the program, such as the length of the forecast time interval, the value of the dead zone, and others, can be easily changed; they are set in the program configuration file.

Using the described programs with two different values of the forecast step, it is possible to predict the price itself, the boundaries of its fluctuations and the trend with sufficiently high accuracy.

Therefore, it is possible to partially automate the trading process, allowing it to send recommendation signals to buy or sell at close values of the trend direction for the selected steps and the results of each of the forecasts.

Keep in mind, that if the forecast is good enough and there is no smoothing, then with a small step (3-5 quanta of discreteness) and a relatively calm market, the forecast deviations from current prices are small, and the forecast chart is very similar to the prices graph, but shifted to the right. As the step increases, this similarity gradually disappears..

The developed program allows us to make calculations, as well as display graphical information in real time.

When making decisions, it is recommended to take into account, not only the results obtained, but also new information and any other stock indicators that are familiar and convenient to the trader.

Due to the use of a hash code, the program works extremely efficiently. Even with small quantum discreteness, the execution time of the program does not impose any restrictions. Therefore, the described predictive indicator, in theory, can find application in high-frequency trading systems.

REFERENCES

1. *Baiynd, Anne-Marie "*The Trading Book: A Complete Solution to Mastering Technical Systems and Trading Psychology*",* McGraw Hill*,* ISBN 9780071766494., *p.272.*
2. *Garimella, Rao Veerabhadra* "A Simple Introduction to Moving Least Squares and Local Regression Estimation",2017, *OSTI 1367799.*
3. *J.B Elsner***,** *A.A.Tsonis* "Singular Spectrum Analysis". A New Tool in Time Series Analysis, 1996, ISBN 978-1-4757-2514-8
4. Enders, Walter, "Applied Econometric Time Series"' New York, Third ed. 2010, ISBN 978-0-470-50539-7.
5. *Mandelbrot, B. B. "*Fractals and Scaling in Finance. Discontinuity, Concentration, Risk" (Selecta, Volume E) Springer-Verlag, New York. 1997, p. 566.
6. *D. Marcek*, "Stock Price forecasting: Statistical, Classical and Fuzzy Neural Network Approach" in MDAI, V. Torra and Y. Narukawa, Eds, vol. 3131. Springer, 2004
7. *Jeffrey S. Smoothing "*Springer Series in Statistics. Methods in Statistics", Hardle W., 2004, ISBN-10:0387947167
8. *John J.Murphy* "Technical Analysis of the Financial Markets. A Comprehensive Guide to Trading Methods and Applications", New York Institute of Finance. 1999.
9. *Max Kuhn Kjell Johnson "*Applied Predictive Modeling", 1st ed. 2013, ISBN-10:1461468485
10. *C.Lento, N. Gradojevic, C. S.Wright* "Investment information content in Bollinger Bands?". Applied Financial Economics Letters. **3** (4), 2007*,* 263–267. doi:10.1080/17446540701206576. ISSN 1744-6546.
11. *M. Alperovich; Y. Alperovich; A. Spiro* "Price and Price Range Prediction in Financial Markets", 2018 Eleventh International Conference Management of Large-Scale System Development (MLSD). IEEE Conferences, 2018. DOI: 10.1109/MLSD.2018.8551778 (Scopus)
12. *Schlossberg, Boris* "Technical Analysis of the Currency Market: Classic Techniques for Profiting from Market Swings and Trader Sentiment", 2006, John Wiley & Sons. ISBN: 9780471973065.